Amendments To The Claims:

1-8. (Canceled)

9. (Currently Amended): Method for processing data, except public key encryption methods based on the composition of finite automates, ~~wherein a Petri net is encoded, written into a memory and~~ the method comprising:

encoding a co-operation of at least two Petri nets, wherein one of the Petri nets is a cryptological data-processing net generated at random,

writing the at least two Petri nets into a memory, wherein the at least two Petri nets are capable of being read and performed from that memory by at least one instance, wherein transitions of [[the]] each Petri net read from at least one tape and/or write on at least one tape at least one symbol ~~symbols or symbol strings,~~ with the aid of at least one head, and

composing the co-operation into a composition result, which ~~wherein data-processing, co-operating nets are composed, the composition result~~ is encoded, written into a memory and read and executed from this memory by at least one instance, wherein the composition result ~~is a net which~~ is equivalent to ~~its components~~ the at least two Petri nets with respect to the external input/output behaviour, except output delays~~, and at least one component is a cryptological component~~.

10. (Currently Amended): Method according to claim 9, ~~characterized in that the components and the composition result are Petri nets,~~ wherein the transitions ~~of the components~~ can receive and send ~~symbols or symbol strings~~ the at least one symbol via ~~optionally existing~~ at least one channel[[s]].

11. (Currently Amended) Method for processing data, the method comprising:

encoding a co-operation of a first Petri net and a second Petri net, wherein one of the Petri nets is a cryptological data-processing net generated at random; ~~characterized in that a Petri net is encoded, written~~

writing the first and second Petri nets into a memory,wherein the at least two Petri nets are capable of being [[and]] read and performed from that memory by at least one instance, wherein transitions of [[the]] each Petri net read from at least one tape and/or write on at least one tape symbols ~~or symbol strings,~~ with the aid of at least one head[[;]] ~~further characterized in that at least one second Petri net, in particular encoded with the properties of the Petri net described in claim 1, is written into a memory and is read and executed from this memory by at least one instance,~~ wherein transitions of [[each]] one Petri net can send the symbols ~~or symbol strings~~ via at least one ~~optionally existing~~ channel~~, which can~~ to be received by transitions of another Petri net [[nets]] via [[this]] the channel ~~or these channels;~~

executing the co-operation ~~further characterized in that the Petri nets are executed~~ by a composition instruction, wherein a third Petri net, equivalent to the ~~co-operating~~ first and second Petri nets with respect to the external input/output behaviour, except output delays, is constituted with the aid of the first and second Petri net;

further characterized in that the Petri nets constitute sequential machines M_Ω with at least one input channel and at least one output channel ~~optionally plural input channels and optionally plural output channels~~, C is a finite set of channels, Δ is a finite set of finite alphabets, γ: $C \to \Delta$, $\Omega = (C, \Delta, \gamma)$ is a communication rule,

$$E_\Omega = \{e \mid e = \{(c,\sigma) \mid \sigma \in \gamma(c) \wedge ((c,\sigma_1) \in e \wedge (c,\sigma_2) \in e \Rightarrow \sigma_1 = \sigma_2)\}\} \cup \{\varnothing\}$$

is a set of input/output events and S is a finite set of states and

$$M_\Omega := \{(S, E_\Omega, \delta, \beta, s_0) \mid \delta : R \to S \wedge \beta : R \to E_\Omega \wedge R \subset S \times E_\Omega$$
$$\wedge (\forall [(s,x), y] \in \beta \forall (c_x, \sigma_x) \in x \forall (c_y, \sigma_y) \in y : c_x \neq c_y) \wedge s_0 \in S\},$$

B with $B \subseteq C$ is a set of internal synchronization channels and the composition $comp_B : M_\Omega^n \to 2^{M\Omega}$ of sequential machines is defined as

$$
\begin{aligned}
comp_B := \Big\{\ & \Big((K_1,...,K_n),\widetilde{K}\Big)\ \Big|\\
& (K_1,...,K_n)=((S_1,E_\Omega,\delta_1,\beta_1,s_{0_1}),...,(S_n,E_\Omega,\delta_n,\beta_n,s_{0_n}))\\
& \wedge \exists T=\Big\{\ ((x_1,...,x_n),(y_1,...,y_n),(s'_1,...,s'_n),\widetilde{x},\widetilde{y})\ |\\
& \quad ([(s_{0_1},x_1),s'_1],...,[(s_{0_n},x_n),s'_n])\in \delta_1\times...\times\delta_n\\
& \quad \wedge([(s_{0_1},x_1),y_1],...,[(s_{0_n},x_n),y_n])\in \beta_1\times...\times\beta_n\\
& \quad \wedge \exists H_x=\bigcup_{i\in\{1,...,n\}} x_i\ \exists H_y=\bigcup_{i\in\{1,...,n\}}\beta_i(x_i):\\
& \quad H_x\in E_\Omega\ \wedge\ H_y\in E_\Omega\\
& \quad \wedge\forall(c,\sigma):\ (c\in B\Leftrightarrow(c,\sigma)\in H_x\cap H_y)\\
& \quad \wedge\widetilde{x}=H_x\setminus H_y\ \wedge\ \widetilde{y}=H_y\setminus H_x\ \Big\}\\
& \exists\widetilde{M}'_\Omega=\Big\{\ \widetilde{K}'\,|\,\exists((x_1,...,x_n),(y_1,...,y_n),(s'_1,...,s'_n),\widetilde{x},\widetilde{y})\in T:\\
& \quad \widetilde{K}'=comp_B([(S_1,E_\Omega,\delta_1,\beta_1,s'_1),...,(S_n,E_\Omega,\delta_n,\beta_n,s'_n)])\ \Big\}:\\
& \widetilde{K}=\Big(\widetilde{S},E_\Omega,\widetilde{\delta},\widetilde{\beta},\widetilde{s_0}\Big)\\
& \quad \wedge\widetilde{S}=(s_{0_1},...,s_{0_n})\cup\bigcup_{(\widetilde{S'},E'_\Omega,\widetilde{\delta'},\widetilde{\beta'},\widetilde{s'_0})\in\widetilde{M}'_\Omega}\widetilde{S'}\\
& \quad \wedge\widetilde{\delta}=\Big\{\ [((s_{0_1},...,s_{0_n}),\widetilde{x}),(s'_1,...,s'_n)]\,|\\
& \qquad ((x_1,...,x_n),(y_1,...,y_n),(s'_1,...,s'_n),\widetilde{x},\widetilde{y})\in T\ \Big\}\\
& \qquad \cup\bigcup_{(\widetilde{S'},E'_\Omega,\widetilde{\delta'},\widetilde{\beta'},\widetilde{s'_0})\in\widetilde{M}'_\Omega}\widetilde{\delta'}\\
& \quad \wedge\widetilde{\beta}=\Big\{\ [((s_{0_1},...,s_{0_n}),\widetilde{x}),\widetilde{y}]\,|\\
& \qquad ((x_1,...,x_n),(y_1,...,y_n),(s'_1,...,s'_n),\widetilde{x},\widetilde{y})\in T\ \Big\}\\
& \qquad \cup\bigcup_{(\widetilde{S'},E'_\Omega,\widetilde{\delta'},\widetilde{\beta'},\widetilde{s'_0})\in\widetilde{M}'_\Omega}\widetilde{\beta'}\\
& \quad \wedge\widetilde{s_0}=(s_{0_1},...,s_{0_n})\ \Big\}.
\end{aligned}
$$

12. (Currently Amended): Method according to claim 9, characterized in that the ~~data-processing~~ Petri nets are constituted by a translation of algorithms.

13. (Canceled).

14. (Currently Amended): Method according to claim 9, characterized in that at least one Petri net ~~component~~ deflates compressed data.

15. (Currently Amended): Method according to claim 9, characterized in that one Petri net ~~component~~ reads data and adds a ~~feature or~~ watermark to the data by changing these data, which is ~~not or only slightly~~ at least partially obstructing the use of these data.

16. (Currently Amended): Method according to claim 9, characterized in that [[an]] at least one encoder and [[a]] at least one combiner of plural input channels are composed into one output channel, wherein the at least one encoder and the at least one combiner are Petri nets, the at least one combiner maps the data received via the input channels on the output channel, and the output of the at least one combiner is the input for the at least one encoder.

17. (Currently Amended): Method according to claim [[9]] 16, characterized in that a decoder and a separator are combined, wherein the decoder and the separator[[,]] ~~respectively, is a~~ are each Petri nets, wherein the decoder [[and]] may be an inversion of [[an]] the at least one encoder ~~and a combiner, respectively, which~~ that has been composed with [[a]] the at least one combiner, wherein the separator may be an inversion of the at least one combiner that has been composed with the at least one combiner, wherein ~~or encoder, respectively, according to the method described in claim 16,~~ the separator maps the data received via the input channel [[on]] or the output channel, and the output of the encoder is the input for the separator.

18. (Currently Amended): Method according to claim 9, ~~characterized in that at least one component is a~~ wherein the cryptological ~~component~~ data-processing net ~~which~~ receives and processes data from a cryptological function which is executed in a protected manner, wherein the composition result does not work or works in an erroneous manner in the case that no data or erroneous data are received from the cryptological function.

19. (Currently Amended): Method according to claim 18, characterized in that a composition is executed omitting the cryptological ~~component~~ data-processing net ~~or at least one of the~~

~~cryptological components~~, wherein the composition result has at least one limitation with respect to usability, compared with the composition result which has been formed without said omission.

20. (Currently Amended): Method for processing data, ~~wherein a Petri net is encoded~~, the method comprising:

encoding a Petri net, which is written into a memory and read and performed from that memory by at least one instance, wherein transitions of the Petri net read from at least one tape and/or write on at least one tape symbols ~~or symbol strings~~, with the aid of at least one head, [[and]]

wherein a cryptological data-processing net ~~or a program, respectively~~, receives and processes ~~second~~ data from a cryptological function which is executed in a protected manner, and the data-processing net ~~or the program, respectively~~, does not work or works in an erroneous manner in the case that no ~~second~~ data or erroneous ~~second~~ data are received, wherein the cryptological function is fixedly attached to the device which executes the data-processing net ~~or the program, respectively~~.

21. (Currently amended): Method according to claim 20, characterized in that a value exceeding ~~the calculation of~~ a first function value of the cryptological function is stored such that it is not readable or changeable [[for]] by an aggressor, and ~~on further calculation of a further function value this~~ the value influences the calculation of a second function value ~~result of the further calculation~~, the value changing according to a predetermined rule.